Microcontrollers embrace fuzzy logic

Most current fuzzy-logic applications run very nicely on general-purpose microcontrollers. For very demanding applications, you can turn to special-purpose fuzzy chips.

Gary Legg, Senior Technical Editor

Electronic designers are getting a pleasant surprise as they begin to apply fuzzy logic to embedded controllers. They expected easier development; they expected simpler implementations. Now they're finding that they seldom need new and unfamiliar hardware; most fuzzy-logic applications run very nicely on familiar microcontrollers.

A few demanding applications do require special fuzzy-logic ICs, and a variety of ICs are now available for those tasks. Some chips target industrial control, others aim at consumer electronics and appliances. Still others take a more general approach, offering speed and versatility that you can apply however you want.

But for most current fuzzy-logic applications, general-purpose microcontrollers are a good match. Their on-chip peripherals and interfaces are essential for many embedded systems, and the addition of fuzzy-logic code doesn't usually impose much of a burden in either processing or memory use. In most cases, the fuzzy code requires less than 1 kbyte of on-chip ROM and considerably less RAM. In the majority of current fuzzy-logic applications, an 8-bit microcontroller is entirely adequate.

An impressive array of software tools is also making fuzzy-logic design relatively painless. There are tools that generate assembly-language code for a variety of popular microcontrollers and for dedicated fuzzy-logic ICs; some also generate C code that you can run on a processor of your choice.

These increasingly sophisticated tools take much of the complicated mathematics out of control-system design and replace it with a much more intuitive approach (**Refs 1** and **2**). Some are even using neural networks that automatically generate fuzzy-logic designs by observing example situations of how you want a control system to behave (see **box**, "Neural nets make fuzzy logic smarter").

General- or special-purpose IC?

Your choice between a microcontroller or a dedicated fuzzy-logic IC depends primarily on speed requirements and the complexity of your system (**Fig 1**). In very simple applications, you can forego fuzzy logic and implement look-up tables, but this approach is practical only for systems that are almost trivial—for example, with two inputs and one output. At the other extreme, applications that require a large number of fuzzy-logic rules or that have stringent speed requirements will require a dedicated fuzzy-logic IC. In between, where most present applications lie, a general-purpose microcontroller running fuzzy logic is the preferred implementation.

Be careful about implementing microcontroller-based fuzzy logic in C code, however; in some cases, execution time can be too long. But don't be too hasty in discounting C for every application. Fuzzy logic isn't very computationally demanding, so even an inefficient implementation will be fast enough in some cases.

The cost difference between a microcontroller and a fuzzy-logic IC isn't terribly significant. Although dedicated fuzzy chips usually cost in the tens of dollars, they can be much cheaper. American Neuralogix's NLX200, for example, sells for around $2 in large quantities. You can get a microcontroller for as little as $1, but you'll usually pay more to get the features you need, such as adequate memory and analog inputs. Will Schreiber, a senior program manager in Intel's Embedded Microcomputer Division, says a suitable 8051 will cost about $3 in volume quantities, and suitable 16-bit controllers will run about $15 to $20.

The microcontroller you choose for fuzzy logic may well be one you're already using for nonfuzzy applications. "You should essentially choose one that has the on-chip peripheral components you need," says Hyperlogic president Fred Watkins. Hyperlogic provides

fuzzy-logic development tools for a variety of controllers, but, according to Watkins, "Almost any controller with a couple of index registers and an accumulator can get the job done."

Resolution and number of bits

You will, however, have to consider what the fuzzy-logic practitioners refer to as resolution. Resolution is akin to precision in that it determines whether you need a microcontroller with 8, 16, or possibly 4 bits. It often depends simply on the number of degrees of membership your application needs (**Refs 1** and **3**)—essentially how "fuzzy" the processing can be and still give acceptable results. Most fuzzy applications require less than 100 degrees of membership, so an 8-bit controller, providing 256, is more than adequate.

You also have to look at your "universe of discourse," another bit of fuzzy jargon. For example, a crane that moves items over large distances has a large universe of discourse; and, because a crane must also move precisely, a fuzzy-logic controller for a crane may need more than 8 bits of resolution. Likewise, the distances covered and the precision required by a disk drive's read/write head may also necessitate use of a 16-bit controller—not only for resolution, but also for speed. In contrast, a fuzzy home-heating system may need only an 8-bit controller, or even a 4-bitter.

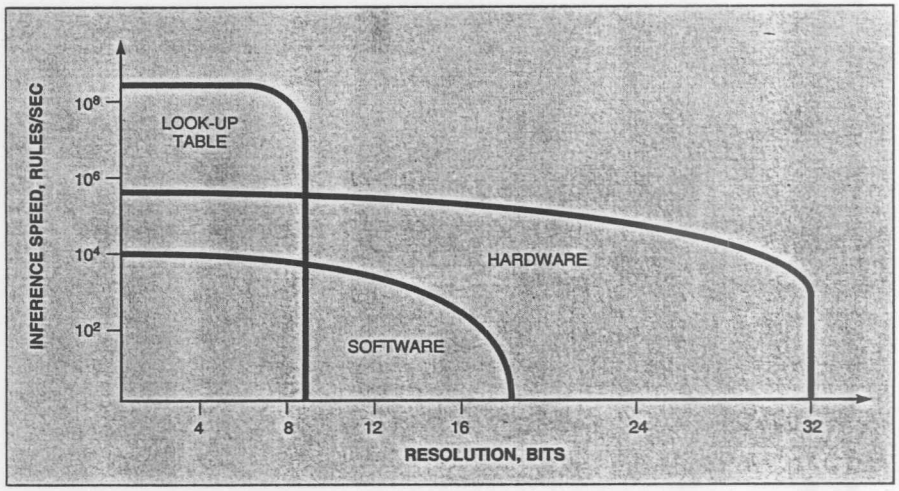

Fig 1—You can implement fuzzy logic in software on microcontrollers or in special-purpose fuzzy-logic ICs. The choice depends on your requirements for speed and resolution. For extremely simple applications, you can just use look-up tables.

How fast a microcontroller needs to be usually depends on how complex your fuzzy system is—primarily how many rules it has. Intel engineer Joe Altnether says if your fuzzy system has more than 25 rules, you'll probably need a 16-bit controller for its speed, even if you don't need its extra resolution. Most fuzzy-logic applications require fewer than 25 rules, however.

Architectures increase speed

Architectural features in some microcontrollers can be helpful in boosting fuzzy logic's speed. A 3-operand compare instruction is nice, according to Altnether, because it lets you perform nondestructive compares—operations that are useful for evaluating the terms of fuzzy-logic rules. A large register file helps, too, Altnether says, because it allows all terms to be within the chip; otherwise, you lose time accessing an external bus. Altnether notes that Intel's MCS96 controllers have both of those features.

In Hitachi's H8/300 and H8/500 microcontrollers, register-rich architectures with indirect and indexed addressing modes help speed up fuzzy logic. The controllers' fuzzy runtime modules, produced by development

Looking ahead

A logical location for fuzzy-logic hardware is inside microcontrollers, as peripheral components. Fuzzy-chip suppliers Togai Infralogic and American Neuralogix both will license their technology for that purpose. Inform GmbH has already put fuzzy firmware in a Siemens microcontroller. Fuzzy "engines" could also find their way into programmable arrays.

Separate fuzzy coprocessors are another approach. Siemens already has one, the 81C99; SGS-Thomson is working on one. The Weight Associative Rule Processor (WARP) from SGS-Thomson will work with the ST9 and ST10 families of microcontrollers, and its technology will also be available for inclusion in custom chips.

Eventually, neural-net hardware will go into fuzzy-logic chips, allowing the development of systems that adapt to changing situations. Both Motorola and National Semiconductor are making long-range plans to incorporate neural nets. Motorola has no timetable; National says it could have neural-net chips in two to three years.

Dedicated fuzzy ICs have yet to find much of a market in the US, but that will change as embedded-controller designers get past their initial simple designs and move on to more complex ones. That scenario is already being played out in Japan, where companies such as Omron and Oki are doing a brisk business in fuzzy chips.

Sensor manufacturers could be the big winners as fuzzy logic catches on. Because fuzzy logic's "intelligence" replaces human intelligence, fuzzy systems will need sources of input data to replace human input. All kinds of sensors—temperature, humidity, pressure, and so forth—will be in greater demand.

tools from Togai Infralogic, make heavy use of those modes in accessing tables. Hitachi claims that benchmark results demonstrate the features' value.

In some speed-critical applications, even some with only a few rules, a fuzzy-logic controller might require a fast processor just to keep up. High-speed motor control is one such example; automotive applications provide others.

Automotive applications require high speed, and fuzzy controllers for cars can be fairly complex. Emdad Khan, the creator of National Semiconductor's NeuFuz fuzzy-logic development tool, says a fuzzy antilock braking system for a car will probably require a 16-bit microcontroller, and fuzzy-logic engine control might even require 32 bits. As

Text continued on page 106

Neural nets make fuzzy logic smarter

You don't have to be a mathematician to design a control system with fuzzy logic, but you do have to know how the system should generally behave. Essentially, you must be able to define a series of rules, in the form of if-then statements, which define that behavior. That's not always possible, though, either because you don't have that kind of expert knowledge or because the control system is too complicated. When a fuzzy system has 20 or so rules, the human mind has difficulty comprehending all the cause-and-effect relationships. Even some fuzzy systems with far fewer rules can be hard to grasp.

Neural networks, patterned somewhat after the human brain, can relieve you of many of the details of fuzzy design. Neural nets are good at finding patterns in masses of data and therefore are good at generalizing fuzzy-logic rules by finding patterns in the input and output data of sample situations. Essentially, neural nets identify the cause-and-effect relationships so you don't have to. This capability is available, to one degree or another, in fuzzy-logic tools from National Semiconductor, Inform Software, and Togai Infralogic. Motorola expects to introduce neural-based tools before the end of the year.

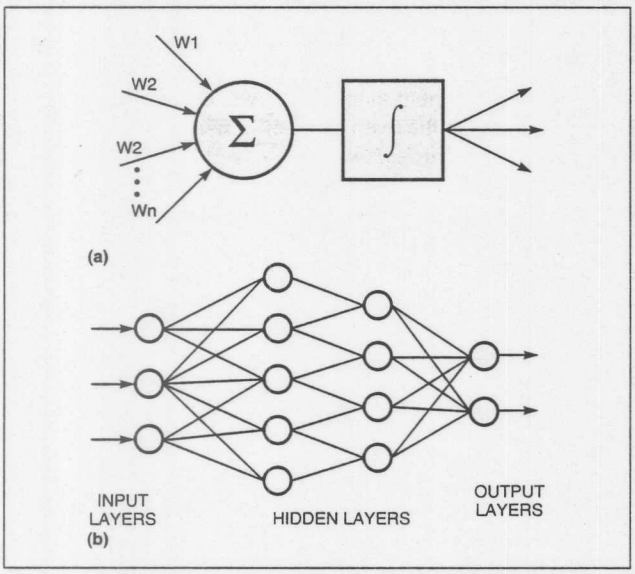

Fig A—Each neuron in a network (a) accepts weighted inputs and produces an output. The network itself (b) consists of layers of neurons, and each connection between neurons also has an associated weight.

Neural networks "learn" and generate rules by processing "training sets"—many examples of desired system behavior that are represented by input and output data. Each neuron in a network (**Fig Aa**) accepts weighted inputs and produces an output. The network itself (**Fig Ab**) consists of layers of neurons, and each connection between neurons also has an associated weight. As the neural net processes the sets of input and output data, it adjusts all its weights until the outputs that it produces closely match the outputs in the training sets. The final weights represent knowledge of desirable system behavior that is translatable to fuzzy-logic rules and membership functions.

You may need to augment the training data for a neural-based fuzzy tool with other information. With National's NeuFuz tool, for example, you specify the number (but not the types) of membership functions you want. NeuFuz then determines what types of membership functions are necessary and generates fuzzy rules. If it comes up with too many rules, you can instruct it to discard those that have the least effect. Then, as part of an iterative development process, you make sure that the system still gets results that are within an acceptable tolerance.

Details of the forthcoming neural tools from Motorola aren't yet available, but the Motorola approach supposedly will differ somewhat from National's. National's NeuFuz, developed by fuzzy-logic expert Emdad Khan, takes an approach referred to as supervised learning. Motorola's tool, developed by fuzzy-logic researcher Chuan-Chang Hung, uses unsupervised learning. The result of each, however, is automatic rule generation.

Automatic generation of fuzzy rules is not, at present, a panacea. Neural networks are computationally demanding, so neural-based tools are somewhat limited in their abilities. National's NeuFuz, for example, can process a maximum of four inputs, although a future version will allow 10 inputs. Running a neural-based tool can also eat up quite a bit of computer time.

Neural nets in hardware

Eventually, though, hardware for accelerating neural computations will become available and make automatic rule generation much more attractive. When that happens, neural nets can start moving from off-line fuzzy-logic development tools into embedded fuzzy controllers. With neural networks inside the controllers, fuzzy systems will be able to adapt to changing situations instead of relying on predetermined rules.

Sources of fuzzy-logic products

For free information on fuzzy-logic products such as those described in this article, circle the appropriate numbers on the postage-paid Information Retrieval Service card or use EDN's Express Request service. When you contact any of the following manufacturers directly, please let them know you saw their products in EDN.

American Neuralogix Inc
Sanford, FL
(407) 322-5608
Circle No. 480

Aptronix Inc
San Jose, CA
(408) 428-1881
Circle No. 481

ByteCraft Ltd
Waterloo, ON, Canada
(519) 746-6751
Circle No. 482

Fuzzy Systems Engineering
San Diego, CA
(619) 748-7384
Circle No. 483

Hitachi America Ltd
Brisbane, CA
(415) 589-8300
Circle No. 484

Hyperlogic Corp
Escondido, CA
(619) 746-2765
Circle No. 485

Inform GmbH
Aachen, Germany
(49) 2408 6091
Circle No. 486

In the US,
Inform Software Corp
Evanston, IL
(708) 866-1838
Circle No. 487

Intel Corp
Santa Clara, CA
(800) 468-8118
Circle No. 488

Integrated Systems Inc
Santa Clara, CA
(408) 980-1500
Circle No. 489

Metus Systems Group
Chappaqua, NY
(914) 238-0647
Circle No. 490

Modico Inc
Knoxville, TN
(615) 531-7008
Circle No. 491

Motorola Inc
Austin, TX
Contact local office
Circle No. 492

National Semiconductor
Santa Clara, CA
(800) 272-9959
Circle No. 493

Oki Semiconductor
Sunnyvale, CA
(408) 720-1900
Circle No. 494

SGS-Thomson Microelectronics
Milan, Italy
(39) (2) 575 461
Circle No. 495

In the US,
Phoenix, AZ
(602) 867-6259
Circle No. 496

Siemens Semiconductor Group
Munich, Germany
(49) 89 4144 8223
Circle No. 497

In the US,
Siemens Corp
Santa Clara, CA
(408) 777-4500
Circle No. 498

Togai Infralogic Inc
Irvine, CA
(714) 975-8522
Circle No. 499

EDN's Super Circle Number
For more information on the fuzzy-logic products available from all the vendors listed in this box, you need only circle one number on the postage paid reader service card. **Circle No. 500**

Life gets fuzzier

When Joe Altnether helps with the laundry, he faces a washer with "a console like a 747's." The Intel fuzzy-logic specialist has no problems operating powerful computers, but the family washing machine leaves him scratching his head.

That could change, though, if American appliance manufacturers follow the lead of their Japanese and European counterparts. In Europe and Asia, where energy is expensive, appliances already incorporate fuzzy-logic control to make them more efficient, and thus cheaper to use. Often, manufacturers add other fuzzy features to simplify operation. Appliances learn to "think" much as a human would, eliminating the need for costly and cumbersome user interfaces.

Rather than depending on a human to instruct it, a washer controlled by fuzzy logic incorporates sensors to gauge conditions. A washer from Matsushita, for example, measures the opacity of the wash water to determine how well the soap has dissolved and how dirty the water is. If the water is fairly clear, the fuzzy controller can specify a shorter wash cycle. In the washer's paddles are load sensors that gauge how much laundry you've put in. If you've put in only a little, the washer can use less water.

Appliance makers in the US have been slow to embrace fuzzy logic, but they're starting. Whirlpool uses fuzzy logic in one of its refrigerators to save energy during the defrost cycle. The fuzzy controller senses temperature changes and defrosts only when necessary, rather than at regular intervals.

You'll be seeing other fuzzy household products, too. Smart vacuum cleaners that detect the difference between a bare floor and a plush rug will adjust brush height and suction force accordingly. Intelligent microwaves will detect moisture and temperature inside the oven in order to defrost and cook your frozen burritos to perfection.

Cars are a natural application for fuzzy logic. Antilock brakes, climate control, and transmission control have already been implemented with fuzzy controllers. Japanese car makers have taken the lead, but American auto producers are very active.

Detroit isn't boasting about fuzzy logic, though; apparently it's not convinced that consumers will view "fuzzy" in a favorable light.

Only Saturn is willing to discuss its fuzzy activities. The Saturn transmission controller, based on a Motorola 68HC11, uses fuzzy logic to assist in braking on downhill grades. The controller accepts information about speed, throttle position, brake-application intensity, and the amount of time braking. With that information, it determines the appropriate gear for downshifting as a car accelerates downhill. The operation is smooth, and the average driver never notices it. It results in less braking effort, however, and thus prolongs brake-system life.

speed requirements increase, however, you'll probably want to consider one of the special ICs that implement fuzzy-logic features in hardware.

Fuzzy chips give blazing speed

If you need blazing speed, then you need a dedicated fuzzy-logic IC (see **box**, "Fuzzy chips turn on the speed"). Dedicated chips are useful not only for speed-critical applications and large rule bases, but also for processing multiple rule bases simultaneously. In a car, for example, one fuzzy chip could control an air bag and antilock brakes and still have capability left over.

Development tools can (and probably should) play a big part in your selection of fuzzy-logic hardware. Not all tools support all chips, and the different tools have different capabilities and different prices. In general, though, fuzzy-logic tools are so capable that developing fuzzy-logic applications without them just doesn't make sense. The tools are good not only for creating a fuzzy system and generating code, but also for helping you get the system working and then tuning it so that it works well in all situations.

The best of the fuzzy development tools provide graphical interfaces that greatly simplify the creation and editing of membership functions. Some also display your application's outputs in a 3-D control surface, so that you can adjust your inputs and quickly see the results.

Some fuzzy tools even let you tune a system while it's running. While you're watching a display of your system's behavior, you can change one or more of the fuzzy membership functions and immediately see the result; there's no need to stop and recompile.

Real-time tuning is more than just a software-development convenience. Bert Hellenthal, a senior application engineer for Inform Software, reports that he's used the real-time capability of Inform's Fuzzytech tools to tune the fuzzy-logic controller for a car's active suspension system. While the car roared around a test track, Hellenthal sat next to the driver and adjusted the controller. "It cuts optimization time down to a fraction," he says.

For some designers, however, tuning a fuzzy system empirically doesn't provide enough assurance of system stability. They want mathematical proof that a fuzzy system will always work. "They're asking questions like, 'How do you look at the equation?'" says Intel's Joe Altnether, "and 'How do you look at a Bode plot?'" Unfortunately, proof of stability isn't forthcoming, although

Fuzzy chips turn on the speed

When fuzzy-logic applications are very complex or require very fast response—in industrial control, for example—a standard microcontroller may not provide the speed you need. In such cases, you can turn to ICs designed specifically for fuzzy logic. Dedicated fuzzy chips are 10 to 100 times faster than general-purpose devices says Yoke Tanaka, an engineer at Togai Infralogic. In fact, claims Tanaka, Togai's chips process fuzzy logic 10 times faster than an Intel 486 can.

Fuzzy ICs take different approaches to achieve high speed. Togai has implemented a RISC processor with an instruction set optimized for fuzzy logic. Other companies include hardware circuits that perform common fuzzy-logic operations, such as max and min compares.

Simple or complex

Fuzzy chips can be fairly simple or quite complex (**Fig A**). Even the simple ones are very fast; the Neuralogix NLX220, for example, samples its four analog inputs at 25 kHz, and has no problem with real-time processing of rules composed of those inputs. Complex chips can be even faster.

Beyond speed, the main differences between simple and complex chips are in how many rules they can handle, how complex the rules can be, and what kinds of fuzzy membership functions they can accommodate. The NLX220, for example, allows only constant-slope membership functions, in contrast with other ICs that are more flexible. This limitation won't necessarily prevent you from implementing an effective fuzzy-logic controller, but it can restrict the way you design that controller. The NLX 220 allows a total of 111 rule terms, or enough for about 25 or 30 typical rules of 3 or 4 terms each. That's enough for most applications, but inadequate for a very elaborate fuzzy system.

A more complex fuzzy IC can process more rules and may have the ability to process more than one fuzzy rule set. The Oki MSM91U112, for example, can handle 128 rules, which can be split into as many as four sets, thus allowing simultaneous operation of four different fuzzy controllers. The chip can process the total of 128 rules 25,000 times/sec. Input membership functions handled by the Oki chip can be shaped like a triangle, a trapezoid, an "S," or a "Z."

RISC for flexibility

Fuzzy ICs implemented as instruction-based processors or coprocessors (as opposed to implementations of fuzzy-processing hardware circuits) offer both speed and flexibility. Togai's RISC-based FC110, which works either as a processor or as a coprocessor, can evaluate 200,000 rules/sec, and can handle as many as 800 rules. Rules and membership functions for an FC110-based fuzzy application reside in off-chip memory, allowing not only the large number of rules, but also membership functions with no restrictions on size or shape.

Fuzzy coprocessors implement fuzzy-logic functions in hardware and still give you the flexibility of using a microcontroller of your choice. The Siemens 81C99 fuzzy coprocessor connects to virtually any 8-bit controller and processes 7.9 million rules/sec. Like the Togai chip, the 81C99 has off-chip knowledge-base storage, and thus nearly unlimited rules and membership functions.

research is addressing the problem.

Nevertheless, fuzzy-logic advocates claim you can prove a fuzzy controller works for all practical purposes. "Stability is a problem," says National's Emdad Khan, "but by extensive testing, you can eliminate most of the problem. And even with conventional nonlinear systems, you cannot guarantee stability." Steve Marsh, director of strategic operations at Motorola's Austin microcontroller division, concurs. "A mathematical analysis of stability can have so many assumptions about a model's behavior," Marsh says, "that all it does is give you a comfortable feeling. It doesn't necessarily prove anything."

Fuzzy logic can, in some cases, actually improve stability in a conventional system. Conventional control systems are often based on differential equations, and those equations have boundary conditions, beyond which the equations are invalid. "If a system steps outside the boundary conditions," Marsh notes, "the system can get into a runaway situation that is easily detectable by some sensor." If the conventional system is augmented by fuzzy logic, Marsh says, the fuzzy part can bring the system back in control. A helicopter's control system can benefit from such a combination, Marsh claims.

But the overwhelming reason for using fuzzy logic is simply that it can make good design sense. Compared with conventional control-system design, your approach can be more intuitive, and your implementation can be simpler and cheaper. Plus, you can use fuzzy logic where it makes sense and stick with conventional techniques for everything else. The two can work together very nicely. EDN

References

1. Legg, Gary, "Special tools and chips make fuzzy logic simple," *EDN*, July 6, 1992, pg 68.
2. Brubaker, David, "Fuzzy-logic basics: Intuitive rules replace complex math," *EDN*, June 18, 1992, pg 111.
3. Conner, Doug, "Designing a fuzzy-logic control system," *EDN*, March 31, 1993, pg 76.

Senior Technical Editor Gary Legg can be reached at (617) 558-4404, fax (617) 558-4470.

Article Interest Quotient
(Circle One)
High 586 Medium 587 Low 588

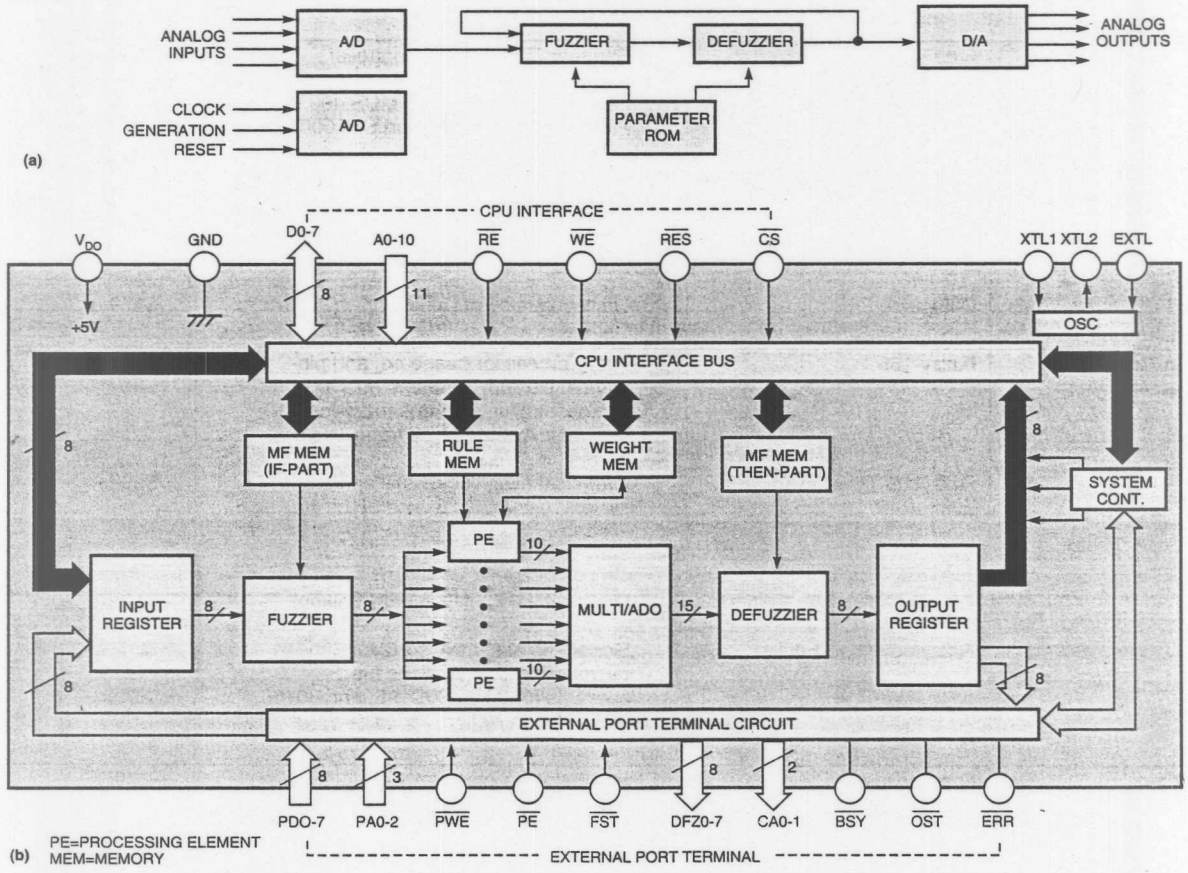

Fig A—A simple fuzzy-logic IC, such as the American Neuralogix NLX220 (a) is ideal for use in appliances. More powerful chips, such as the Oki fuzzy processor (b), are useful for advanced applications such as industrial control.

Table 1—Fuzzy-logic products

Source	Product	Description	Price
American Neuralogix Inc Circle No. 480	NLX220	16-pin fuzzy-logic IC holds rules and membership functions in mask ROM. Holds over 100 rule terms, has four analog inputs and four analog outputs. Also available as core element for ASICs.	$2 (OEM)
	ADS 220	PC-based development system has simulator and parameter generator.	$395
Aptronix Inc Circle No. 481	Fide	Graphical development tool for Motorola microcontrollers. Generates fuzzy code 6805, 68HC05, and 68HC11; later this year will support 68HC08, 68HC16, 68020, 68030, 68040, and generate C code.	$1995 to $3495
ByteCraft Ltd Circle No. 482	Fuzz-C Preprocessor	PC-based tool translates fuzzy-logic descriptive text to C code.	$149
Fuzzy Systems Engineering Circle No. 483	Manifold Editor	Edits rules in a matrix display, provides display of fuzzy sets.	$295
	Manifold Graphics Editor	Displays rules and fuzzy sets; displays designs in 3-D map and slice formats.	$495
Hitachi America Ltd Circle No. 484	Fuzzy Logic Embedded Controller Development Package	Same as Togai's microcontroller development package. Generates code for H8/300, H8/500, or HMCS 400. Future version will support SH7000.	$1530
Hyperlogic Corp Circle No. 485	Cubicalc	Graphical fuzzy-logic development system.	$495
	Cubicalc-RTC	Cubicalc superset; generates C code.	$795
	Cubiquick	Limited version of Cubicalc.	$179
Inform GmbH Circle No. 486	Fuzzy-166	Fuzzy processor based on, and pin-compatible with, Siemens 80C166 microcontroller. Includes fuzzy-logic firmware. Available in Europe.	NA
	Fuzzytech Standard Edition	Graphical fuzzy-logic development system. Generates 8- and 16-bit C code.	$999
	Fuzzytech Explorer Edition	Limited version of Standard Edition.	$199
	Fuzzytech MCU Edition	Generates assembly code for microcontrollers. Versions are available for 8051, 80C196, and 80166.	$1790 to $2090
Intel Corp Circle No. 488	Fuzzybuilder MCU-96	Includes Projectbuilder with Inform's Fuzzytech MCU Edition for 80C196.	$2096
	Fuzzybuilder Explorer 96	Includes Projectbuilder with Inform's Fuzzytech Explorer Edition.	$396

Source	Product	Description	Price
Integrated Systems Inc Circle No. 489	RT/Fuzzy Module	Development system generates real-time fuzzy-logic code in C or Ada.	$5000
Metus Systems Group Circle No. 490	Metus	Fuzzy-logic development and simulation system for embedded applications.	$1250
Modico Inc Circle No. 491	Fuzzie 1.8	PC-based fuzzy-logic shell generates C and Fortran code.	$289
Motorola Inc Circle No. 492	Fide	Version of Fide from Aptronix generates fuzzy code for 6805, 68HC05, and 68HC11. Later this year, will support 68HC08, 68HC16, 68020, 68030, and 68040.	$1495
	Fuzzy Logic Educational Kit	PC-based tutorial includes limited version of Fide. Generates code for 68HC05 and 68HC11.	$195
	Fuzzy Logic Introductory Kit	Fuzzy Logic Educational Kit without code generation.	$68
National Semiconductor Corp Circle No. 493	NeuFuz4 Development Kit	Fuzzy-logic development system uses neural networks for automatic generation of rules and membership functions for COP8 microcontrollers.	$3975
	NeuFuz4 Learning Kit	Limited version.	$199
Oki Semiconductor Circle No. 494	NSM91U112	Fuzzy coprocessor interfaces to most microcontrollers. Provides fast processing of as many as four different rule bases. Development tools from Oki and Togai. Available in Japan.	NA
SGS-Thomson Microelectronics Circle No. 495	Fuzzytech ST62 Explorer	Version of Inform tool for ST6 family. Versions available later this year Versions available later this year will support ST9 and ST10.	NA
Siemens Corp Circle No. 497	81C99	20-MHz fuzzy coprocessor interfaces to most 8-bit microcontrollers. Processes most 8-bit microcontrollers. Processes 7.9 million rules/sec. Supported by software from Inform.	$32
Togai Infralogic Inc Circle No. 499	FC110	8-bit RISC processor optimized for fuzzy logic. Processes large rule bases. Allows user-defined membership functions of arbitrary shape. A core module is available for use in custom ICs.	$45
	TILShell development package	Includes TILShell graphical shell and generates C code. Includes simulation and debugging facilities.	$1899
	Microcontroller development package	Includes TILShell graphical shell and generates assembly code for specific microcontrollers. Versions are available for 8051, 68HC11, and Hitachi's H8/300, H8/500, and HMCS400.	$1499